AD-A102 990 INSTITUTE FOR SOFTWARE ENGINEERING PALO ALTO CA F/G 9/2
DESIGN SCIENCE CAPACITY ASSESSMENT METHODOLOGY.(U)
OCT 78 L M TRAISTER N00014-78-C-0768
UNCLASSIFIED NL

DESIGN SCIENCE

CAPACITY ASSESSMENT METHODOLOGY

INTERIM REPORT 1 - ITEM 001AA

Prepared for:

OFFICE OF NAVAL RESEARCH

CONTRACT NO. N00014-78-C-0768

Date: October 25, 1978

Prepared by:

Leon M. Traister

Institute for Software Engineering
P.O. Box 637, Palo Alto, CA 94303

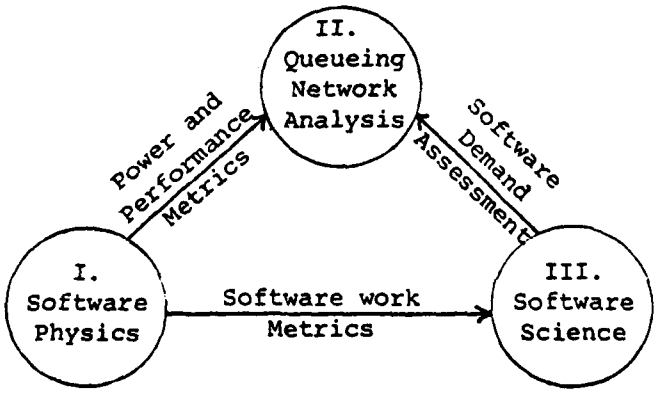
SECTION I

INTRODUCTION

This report presents the outlines of two of three forthcoming papers in the hardware/software design science. The first, "*Foundations and Concepts of Software Physics*", is a new, concise exposition of Kolence's software physics which defines the metric quantitites for both hardware and software capacity and performance. The second paper, "*Queueing Network Analysis in Software Physics*", presents an integration of operational queueing network methodologies into software physics, thus giving effective modeling procedures based on organic properties of hardware and software performance. A substantial feature of this approach is that properties of hardware, configuration and software are not confounded in the analysis, but are kept distinct, finally combining in the model algorithms themselves.

A third paper whose content is under current investigation addresses the feasibility of estimating software work demand using the theory of software science developed by Halstead. Such capability would permit the translation of the content of algorithms into estimates of demand on equipment. This is, of course, a critical input to the performance modeling process, no matter by what means.

The interrelation of the three papers is diagrammed below.



Submission for publication of the first two papers is expected by the early months of 1979. Submission of the third paper is expected in the spring or summer of the same year.

SECTION II

OUTLINE OF THE PAPER

"Foundations and Concepts of Software Physics"

Roger P. Kovach
Institute for Software Engineering

I. ABSTRACT

Software physics is an analytic theory of computer systems performance and capacity which reveals the relationships among work, system times, equipment, configuration and software power, workload characteristics and capacity. It provides a sound uniform basis for all issues pertaining to the management of computing capacity. Kolence's original publication of the theory (KOLE76) set forth the basic definitions and concepts, developed some of the important consequences and dealt with a variety of pragmatic matters in applications of the theory, problems of measurement, and suggested several computational techniques. This paper presents a more explicit statement and development of the logical foundations of the theory, the basic concepts and a more rigorous derivation of results, including some additional conclusions.

II. BACKGROUND

The objective is to develop a theory of executing computing systems. The implications of this specific focus are discussed as well as the distinctions between a theory arising from the properties of physical systems as opposed to the approach of treating those systems as models of mathematical systems. Motivation is provided in a discussion of the pragmatic problems that arise in the diverse areas of capacity management and how a theory embracing all of them resolves many of the difficulties.

III. LOGICAL STRUCTURE OF SYSTEMS

The conventional configuration diagram is viewed as the graph union of paths appearing in instantaneous descriptions. The path graphs are "tree-like" in that they have the upper lattice property. Associated with each device (node) is a set of properties. The covering or containing property of a lattice pertains to those properties. A subset of a graph may be selected by taking all sub-graphs containing a designated property. A rule of composition of these sub-graphs into a logical subconfiguration is developed.

IV. BASIC SYSTEMS

There are several usual methods of forming logical subconfigurations. Every elementary action or set of actions involves a hardware constituent and a software constituent. Hardware may be subsetted by categories, called equipment classes, or by the topology of the configuration, called subconfigurations. The software constituent may be decomposed into a variety of software units. These three basic systems are defined and a notation for representation given.

V. BASIC PROPERTIES - WORK

Work is performed when a change of state occurs. In this setting, software work is performed when the symbol state of a storage container is changed. This fundamental concept has a number of characteristics that make it rich in implications for both theory and application. Among them are invariance under a variety of transformations and extensiveness. A quantity of work may be associated with any logical subconfiguration, at any level, and the flow of work over the system may be described. A functional notation is given for representing these.

VI. BASIC PROPERTIES - TIME

There are a number of times that arise naturally out of system activity and structure. Failure to observe the distinctions among them leads to serious confusion. Each logical subconfiguration has its own execution time, which is the time during which any component of that subconfiguration is active. Thus, every subconfiguration has its own "clock." Activity at one level of subconfiguration against the clock of a containing level gives rise to the definition of elapsed time. The ratio of the amounts of those times defines a family of relative utilizations. Notation and some identities are presented. There is also a distinction between execution time and busy time made which gives rise to a definition of delay time.

VII. DERIVED PROPERTIES - POWER

Power is the rate at which work is done with respect to time. Since work may be associated with any level of subconfiguration and time with that of any containing subconfiguration, a group of power measures are defined. When the work and time are measured at the same level, absolute power is measured. All of the others are relative powers. The relationship among absolute power, relative power and utilizations are developed. Power provides the natural conceptual foundation for definitions of capacity and throughput and other performance characterizations.

VIII. DERIVED PROPERTIES - MP LEVELS

An important measure of the effectiveness of a configuration with respect to a given workload is the degree of concurrency, called an MP level, achieved. This measure is closely related to relative powers and utilizations. These are shown and some significant identities derived.

X. THROUGHPUT POWER VECTORS

Introduction of the appropriate time ordinate transforms work vectors into vectors of relative powers. It is shown that these are, in fact, throughput characterizing vectors.

XI. CAPACITY CALCULATIONS

A family of idealized powers is defined and capacity is defined in terms of them. System capacities may then be calculated.

XII. SUMMARY

Software physics, as far as it is developed here, provides sound concepts for defining, measuring and operationally testing definitions and relationships among work, power, service times, utilization rates. throughput rates and capacities. These arise naturally from the observed properties of the observed systems.

SECTION III

OUTLINE OF THE PAPER

"Queueing Network Analysis in Software Physics"

L. M. Traister
Institute for Software Engineering

I. ABSTRACT

Queueing network models have proven to be effective tools in the analysis of computing system performance. Recent advances in operational methods by Buzen and others have further enhanced their practical use. Kolence's software physics robustly addresses the problem of appropriate metrics for both performance and capacity of devices and configurations. This paper integrates methods and laws of queueing network analysis into an extension of software physics. A substantial feature of this approach is that properties of hardware, configurations and software are not confounded in the analysis but are kept distinct, finally combining in the model algorithms themselves. All basic objectives and assumptions are stated and the basic quantities are defined. The fundamental operational laws are derived. Finally, the concepts are illustrated with both general queueing network examples and the presentation of an actual design case study.

II. BACKGROUND

This section will discuss the features of certain queueing network methodologies, both stochastic modeling and the operational approach more recently developed by Buzen (DENN78). The motivation is then presented for the development of an integrated conception which is both effective and organic to computing system capacity and performance quantities. This conception is an extension of software physics.

III. PRINCIPLES AND ASSUMPTIONS

This section will first discuss the basic requirements which the integrated methodology must meet, e.g., operational vertification of assumptions and measurement of performance quantities. The specific assumptions that are supportive of and supplemental to these requirements are then presented, e.g., job flow balance, device homogeneity and so on.

IV. BASIC QUANTITIES

The quantities of software phsyics which play direct role in the laws and relationships to be subsequently shown are presented. Where direct analogy exists between a software physics quantity and that of the more traditional queueing network analysis, it is noted and a law of transformation given. Examples of such quantities are software work, execution time, absolute and relative powers, etc.

V. LAWS AND RELATIONS

Under the assumptions of Section III, operational laws are derived which relate the basic quantities and formulate additional performance quantities. An example of such a formulation is the fundamental Little's Law expressing response time in terms of throughput power and software work in the system.

VI. APPLICATION

This section will demonstrate the application of the previously derived quantities to the analysis of some general queueing networks. The concepts of device saturation and system bottlenecks are introduced and it is shown how these limit performance. The expressions for the throughput or response time asymptotes and Kleinrock's (KLEI76) point of system saturation are developed in software physics quantities. Finally, the conditions are discussed under which we may proceed to more complete solutions of the queueing network models.

VII. A CASE STUDY

This section will present as an example the application of the above methodology to an actual network design problem recently performed.

VIII. CONCLUSION

This section will briefly summarize the major results. Special emphasis will be given to identification of topics for further investigation and development which will enrich the theory and expand the methodology.

REFERENCES

DENN78 Denning, P.J. and Buzen, J.P., *"The Operational Analysis of Queueing Network Models"*, Computing Surveys, Vol. 10, No. 3 (September, 1978), pp. 225-261.

KLEI76 Kleinrock, L., Queueing Systems, Vol. II, Wiley (1976).

KOLE76 Kolence, K. W., An Introduction to Software Physics, Institute for Software Engineering, Palo Alto (1976).


FILME